APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: APPLICATION FRAMEWORK

APPLICANT:

WERNER AIGNER
Am Groitlfeld 30a
93497 Willmering
Germany

JOERG BERINGER
Wildenbruchstr. 49
60431 Frankfurt
Germany

STEFAN MUELLER
Am Hasenpfad 11B
74889 Sinsheim
Germany

SHAI AGASSI
209 Hollywood Avenue
Los Gatos, California 95032

DENNIS B. MOORE
2856 Hillside Drive
Burlingame, CA 94010

JUERGEN HAGEDORN
Albert Schweitzer Ring
3B 69226
Nussloch
Germany

UDO WAIBEL
3475 Deer Creek Road
Palo Alto, CA 94304

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 327614805 US

09/08/2003
Date of Deposit

APPLICATION FRAMEWORK

RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/456,697, filed on March 21, 2003.

BACKGROUND

[0002] Enterprise solutions have recently focused on allowing disparate components in a heterogeneous information technology (IT) environment to work with each other to implement a business process. This has given rise to the exchange infrastructure (XI), which integrates heterogeneous components, such as customer relationship management (CRM), human resources management (HRM), and product life-cycle management (PLM), in an IT landscape.

SUMMARY

[0003] In one general aspect, a framework for a composite application — that is, an application built on other applications — includes an object access layer, a service layer, and a user interface layer. The object access layer is operable to exchange data with a plurality of enterprise base systems and to present the data to a composite application through a uniform interface. The service layer is operable to provide services to the composite application. The user interface layer is operable to provide user interface patterns that facilitate information exchange between the composite application and a user.

[0004] In certain implementations, a composite application may include business objects, business services, and business processes. A business service may be an action performed on a business object, and a business process may be a combination of business services.

[0005] In particular implementations, a composite application framework includes a database for composite application data, wherein an object access layer is operable to provide local persistency in the database. In these implementations, the object access layer may be operable to provide data synchronization and replication of enterprise base system data in the database.

[0006] In certain implementations, a service layer includes a collaboration services module operable to provide a collaboration service to a composite application and a workflow services module operable to provide a workflow to a composite application. A collaboration services module may be operable to link a semantic object to a business object of a composite application. A workflow may include templates, workflow patterns, and actions, a template describing a workflow, workflow patterns describing portions of the template, and actions executing functions to carry out the workflow patterns.

[0007] In particular implementations, a service layer further includes a container for composite application services, the container operable to provide interfaces for non-framework-generated code.

[0008] In some implementations, a user interface layer includes a user interface framework that separates the user interface elements from the composite application so that the user interface is decoupled from the logic.

[0009] In certain implementations, a composite application framework includes a business object modeler and a business object generator. The business object modeler may be operable

to provide a user interface for constructing a business object, and the business object generator may be operable to generate an executable version of the modeled business object.

[0010] In some implementations, a business object modeler may include an object modeler and a relation modeler. Also, a business object generator may include a generator framework and a persistency generator. Additionally, the business object generator may be operable to code a business object template with metadata and relation data for a business object to generate an executable version of the modeled business object. Furthermore, a business object generator may be operable to generate tables and proxies for a business object.

[0011] Another general aspect includes a method for implementing a composite application in a framework. The method may include generating executable code for a composite application, exchanging data with a plurality of enterprise base systems, presenting the enterprise base system data to the composite application through a uniform interface, and facilitating a user's interaction with the composite application through user interface patterns. A composite application may include business objects, business services, and business processes, wherein a business service is an action performed on a business object, and a business process is a combination of business services. The method may be implemented by hand, machine, an article including a machine-readable medium storing instructions, or otherwise.

[0012] In particular implementations, generating executable code for a composite application includes coding a template with business object metadata and relation data. In some of these implementations, generating executable code further includes generating tables and proxies for a business object.

[0013] Certain implementations may include providing local persistency in a database for composite application data. These implementations may include providing data synchronization and replication of enterprise base system data in the database.

[0014] Particular implementations may include providing a collaboration service to a composite application and providing a workflow to a composite application.

[0015] Certain implementations may include providing a container for composite application services, the container operable to provide interfaces for non-framework-generated code portions.

[0016] Some implementations may include providing user interfaces to model a composite application, the user interfaces allowing specification of attributes and relations for a business object of the composite application. These implementations may further include generating metadata for the business object and relations based on the specifications.

[0017] In another general aspect, a framework for developing and implementing a composite application includes a database, an object access layer, a service layer, a user interface layer, a business object modeler, and a business object generator. The database is operable to store composite application data, and the object access layer is operable to exchange data with a plurality of enterprise base systems, present the data to a composite application through a uniform interface, provide local persistency in the database, and provide data synchronization and replication of enterprise base system data in the database. The service includes a collaboration services module operable to provide a collaboration service to the composite application and a guided procedure services module operable to provide a guided procedure to the composite application. The user interface layer is operable to provide user interface

patterns for displaying information relating to the composite application, and includes a user interface framework that separates the user interface elements from the composite application so that the user interface is decoupled from the logic. The business object modeler is operable to provide a user interface for constructing a business object of the composite application, and the business object generator is operable to generate an executable version of the modeled business object. A composite application may include business objects, business services, and business processes, wherein a business service includes an action performed on a business object, and a business process includes a combination of business services.

[0018] In particular implementations, a business object modeler may include an object modeler and a relation modeler. Furthermore, a business object generator may include a generator framework and a persistency generator. A business object generator may be operable to code a business object template with metadata and relation data for a business object to generate an executable version of the modeled business object and to generate tables and proxies for a business object.

[0019] In certain implementations, a collaboration services module is operable to link a semantic object to a business object of the composite application.

[0020] In some implementations, a guided procedure includes templates, workflow patterns, and actions, a template describing a guided procedure, workflow patterns describing portions of the template, and actions executing functions to carry out the workflow patterns.

[0021] In particular implementations, a service layer further includes a container for composite application services, the container operable to provide interfaces for non-framework-generated code.

[0022] Details of one or more implementations are set forth in the accompanying drawings and the description below. Other features and advantages may be apparent from the description and drawings, and from the claims.

DRAWING DESCRIPTIONS

[0023] These and other aspects will now be described in detail with reference to the following drawings.

[0024] FIGs. 1A-B are block diagrams illustrating an example integrated enterprise management system.

[0025] FIG. 2 is a block diagram illustrating an example framework for a composite application.

[0026] FIG.3 is a block diagram illustrating another example framework for a composite application.

[0027] FIG. 4 is a block diagram illustrating design-time components for a composite application framework.

[0028] FIGs. 5A-I are user interfaces that illustrate an implementation of a business object modeler.

[0029] FIG. 6 is a block diagram illustrating run-time components for a composite application framework.

[0030] FIGs. 7A-B are block diagrams illustrating components for guided procedures for a composite application framework.

[0031] FIGs. 8A-K are user interfaces that illustrate an implementation of a guided procedure.

[0032] FIG. 9 is a user interface illustrating collaboration services.

[0033] FIG. 10 is a block diagram of a user interface framework for a composite application framework.

[0034] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0035] The systems and techniques described here relate to a framework for developing and implementing applications in an enterprise management system. For example, a framework may be used to develop and implement a composite application, which overlays an enterprise IT platform and uses it to implement processes that are not the core enterprise transactional processes. That is, a composite application may orchestrate a business process in synchronization with existing processes (e.g., native processes of enterprise base systems) and leverage existing investments in the IT platform. Furthermore, composite applications may be run on a heterogeneous IT platform. In doing so, composite applications may be cross-functional. That is, they may drive business processes across different applications, technologies, and organizations. Accordingly, composite applications may drive end-to-end business processes across heterogeneous systems. Additionally, composite applications may be combined with each other in order to enlarge the process coverage. Composite applications may also support semi-structured processes, tackle event-driven and knowledge-based scenarios, and support a high degree of collaboration in teams. In teams, for example, people may work on specific tasks in specific roles in specific teams. Composite applications may relate knowledge, structured information, and/or unstructured information within the context of a business process and may be triggered by events, aggregate and contextualize information,

and drive collaboration and transactions. Different applications supported by different frameworks may have any combination of these characteristics. Thus, different implementations of the framework may be used for developing and implementing various types of applications.

[0036] FIGs. 1A-B illustrate an example integrated enterprise management system 100. In system 100, clients 110 access data over a communication network 120 through a portal 130. Network 120 may be any appropriate type of communication network, such as, for example, a local area network (LAN), a wide area network (WAN), an enterprise network, a virtual private network (VPN), the Internet, and/or the Public Switched Telephone Network (PSTN). Clients 110 may be any machines or processes capable of communicating over network 120. In particular implementations, clients 110 may be Web Browsers and, optionally, may be communicatively coupled with network 120 through a proxy server.

[0037] Portal 130 provides a common interface to program management services. In operation, portal 130 receives requests from clients 110 and generates information views 131 (e.g., Web pages) in response. The portal may implement a user-role based system to personalize the common interface and the information views 131 for a user of one of clients 110. A user may have one or more associated roles that allow personalized tailoring of a presented interface through the generated information views 131.

[0038] In particular implementations, the portal may includes a security component, a content directory component, a view builder, a content management component, and one or more service interfaces to an enterprise management consolidation system 140. The security component may protect data transmission using encryption (e.g., Secure Socket Layers (SSL)),

digital signatures, and/or watermarking. The view builder may create role-based interactive views (e.g., Web pages) for presentation to users. The content management component may include a retrieval and classification component (e.g., Text Retrieval and Extraction (TREX) component) and a collaboration component. The retrieval and classification component may automatically analyze unstructured components to identify know-how. The service interfaces could include an Internet Transaction Server (ITS) component, various connectors, such as a Java Connector, and a Business Intelligence platform.

[0039] Portal 130 communicates with enterprise management consolidation system 140, which consolidates multiple application services. Portal 130 receives information 141 from enterprise management consolidation system 140 for use in fulfilling the requests from clients 110. Enterprise management consolidation system 140 provides integrated application services to manage business objects and processes in a business enterprise. The business objects and processes may be resources (e.g., human resources), development projects, business programs, inventories, clients, accounts, business products, and/or business services.

[0040] Enterprise management consolidation system 140 communicates with enterprise base systems 150 to obtain multiple types of information 151. Enterprise base systems 150 may include various existing application services, such as customer relationship management (CRM) systems, human resources management (HRM) systems, financial management (FM) systems, project management (PM) systems, knowledge management (KM) systems (e.g., documents attached to a business object), business warehouse (BW) systems, time management (TM) systems, and/or electronic file and mail systems. The enterprise base systems may also

include an integration tool, such as an exchange infrastructure (XI), which provides another level of integration among base systems.

[0041] Enterprise management consolidation system 140 may consolidate and integrate the data and functionality of enterprise base systems 150 into a single enterprise management tool. This enterprise management tool may include systems and techniques to facilitate creation and execution of new applications within the enterprise management consolidation system. These new applications may be composite applications and may readily draw on the resources of enterprise base systems 150 to cross over traditional enterprise application boundaries and to handle new business scenarios in a flexible and dynamic manner, allowing rapid and continuous innovation in business process management. A virtual business cycle may be created using such cross-functional applications, where executive-level business strategy may feed management-level operational planning, which may feed employee-level execution, which may feed management-level evaluation, which may feed executive-level enterprise strategy. The information generated at each of these stages in the enterprise management cycle may be readily consolidated and presented by the enterprise management consolidation system 140 using customized composite applications. The stages may provide and consume determined services that may be integrated across multiple disparate platforms.

[0042] Portal 130, enterprise management consolidation system 140, and enterprise base systems 150 may reside in one or more programmable machines, which may communicate over a network or one or more communication busses. For example, base systems 150 may reside in multiple servers connected to an enterprise network, and portal 130 and enterprise management consolidation system 140 may reside in a server connected to a public network. Thus, system

100 may include customized, Web-based, composite applications, and a user of the system may access and manage enterprise programs and resources using these customized, Web-based, composite applications from anywhere that access to a public network is available.

[0043] FIG. 1B further illustrates enterprise management consolidation system 140 for the example. System 140 includes a persistence layer 142 and one or more base system connectors 145. Base system connectors 145 enable data exchange and integration with enterprise base systems. Base system connectors 145, for example, may include an Enterprise Connector (EC) interface, an Internet Communication Manager/Internet Communication Framework (ICM/ICF) interface, an Encapsulated PostScript® (EPS) interface, and/or other interfaces that provide Remote Function Call (RFC) capability.

[0044] Persistence layer 142 provides enterprise management consolidation system 140 with its own database 143 and data object model 144. Database 143 and object model 144 provide a consolidated knowledge base to support multiple enterprise management functions, such as portfolio management, project execution, risk assessment, budgeting, scheduling, workforce planning, skills management, business forecasting, and capacity modeling, which could all be could be created as composite applications 149. Active communication between persistence layer 142 and the base systems may provide a tight linkage between real-time operational data from multiple base systems and an integrated enterprise analysis tool to allow strategic enterprise management and planning.

[0045] Data object model 144 may represent a subset of data objects managed by the base systems. That is, not all of the data aspects tracked in the base systems need to be recorded in data object model 144. Data object model 144 may have defined relationships with data

objects stored in the base systems; for example, certain objects in data object model 144 may have read-only or read-write relationships with corresponding data objects in the base systems. These types of defined relationships may be enforced through the communication system built between persistence layer 142 and the base systems. Thus, persistence layer 142 may be used to effectively decouple application development and execution from the underlying base systems.

[0046] Applications 149 take advantage of this decoupling from back-end systems to flexibly integrate existing systems and new functional components into business processes. Furthermore, the applications may drive business processes across different platforms, technologies, and organizations. Additionally, applications 149 may support semi-structured processes, aggregate and contextualize information, handle event-driven and knowledge-based scenarios, and support a high degree of collaboration in teams, including driving collaboration and transactions.

[0047] Applications 149 may be created using a set of tools that enable efficient application development. The tools may enable efficient application development by providing application patterns that support model-driven composition of applications in a service-oriented architecture.

[0048] An object modeling tool 146 enables creation of new business objects in the persistence layer 142 by providing a mechanism to extend data object model 144 dynamically according to the needs of an enterprise. A process modeling tool 147 enables creation of new business workflow and ad hoc collaborative workflow. A user interface (UI) tool 148 provides UI patterns that may be used to link new objects and workflow together and generate

standardized views into results generated by applications 149. Object modeling tool 146, process modeling tool 147, and UI tool 148, thus, may be used to build the components of applications 149 to implement new enterprise management functions without requiring detailed coding activity.

[0049] Process modeling tool 147 may include guided procedure templates with pre-configured work procedures that reflect best practices of achieving a work objective that is part of a larger cross-functional application scenario. Such a work procedure may include contributions from several people, creation of multiple deliverables, and milestones/phases. Moreover, whenever an instantiated business object or work procedure has lifetime and status, the progress and status of the object or work procedure may be made trackable by the process owner or by involved contributors using a dashboard that displays highly aggregated data. A dashboard and a page that provides access to status information about ongoing work, such as a personalized work place, may be two UI patterns that are provided by UI tool 148.

[0050] Moreover, there may be other UI personalizations. For example, if there is a concept of personalized items, such as, for example, objects, recent objects, related objects, or preferred objects, then an object picker UI pattern, provided by UI tool 148, may be included to let users pick their favorite object directly.

[0051] If people are to be searched for, either for choosing one individual person or for generating a collection of people meeting some criterion, a people finder concept may be applied. A key aspect of searching for a person may be described as an attribute within the user's activity, qualification, interest, and/or collaboration profile. For a given composite

application scenario, people collections may be stored as personal or shared collections using the people finder to make them available for further operations later on.

[0052] If there is a strategic view on a composite application scenario, analytics of the overall portfolio may be made available in the form of a collection of UI components. A view selector may be used to display/hide components, and a component may be toggled between graphical and numerical display and may include a drop-down list or menu to select sub-categories or different views.

[0053] Composite application scenarios may provide related information to the user when possible, and some parts within a larger application scenario may define what kind of related information is to be offered. Heuristics may be used to identify such relatedness, such as follows: (1) information that is related to the user due to explicit collaborative relationships such as team/project membership or community membership; (2) information that is similar to a given business object in a semantic space based on text retrieval and extraction techniques; (3) recent objects/procedures of a user; (4) other people performing the same or similar activity (e.g., using same object or procedure template having the same workset); (5) instances of the same object class; (6) next abstract or next detailed class; (7) explicit relationships on the organizational or project structure; (8) proximity on the time scale; (9) information about the underlying business context; and/or (10) information about the people involved in a collaborative process.

[0054] Composite applications also may include generic functionality in the form of ControlCenter Pages that represent generic personal resources for each user. These may refer to the following pages where appropriate: (1) MyOngoingWork page: provides access to status

information about ongoing work of a user (Ongoing work may refer to the state of business objects as well as guided procedures); (2) MyDay page: lists today's time-based events that are assigned or related to a user; (3) MyMessageCenter page: displays pushed messages and work triggers using a universal inbox paradigm with user selected categorical filters; and/or (4) MyInfo: provides access to personal information collections (e.g., documents, business objects, contacts) including those located in shared folders of teams and communities of which the user is a member. The page may also provide targeted search in collaborative information spaces such as team rooms, department home pages, project resource pages, community sites, and/or personal guru pages.

[0055] FIG. 2 illustrates a framework 200 for a composite application. In general, framework 200 leverages and enhances underlying enterprise base systems 290, which could include an XI, supporting business transaction systems such as CRM, HCM, and PLM, Knowledge Management Warehouse (KW), and BW, with tools, content, and guidelines to provide a foundation for developing and executing composite applications.

[0056] As discussed previously, composite applications typically implement new or additional processes, as opposed to the core transactional processes, in an existing IT landscape. Composite applications may also support semi-structured processes, tackle event-driven and knowledge-based business scenarios, and support collaboration in teams. In particular implementations, composite applications may support the Java stack.

[0057] As illustrated in FIG. 2, framework 200 includes an object access layer (OAL) 210, a service layer 220, a user interface (UI) layer 230, and a metadata repository 240. OAL 210 manages interaction between composite applications and enterprise base systems 290. In doing

so, OAL 210 provides a uniform interface for composite applications. Thus, OAL 210 reduces the knowledge needed for a composite application developer about the source of data because OAL 210 sits on top of and embraces different connectivity technologies. Coding and configuration data for OAL 210 may be automatically generated, at least in part, by business object metadata in repository 240. Furthermore, OAL 210 allows for local persistency (e.g., connectivity to a local database such as an application database 250 to store data). Data synchronization and replication of remote data (e.g., data in back-end systems) into the local persistency database may be supported. For an application sitting on top of layer 210, the source of the data may be completely transparent, which may assist in keeping application logic stable since the application is, at least for the most part, not affected by underlying systems. In some implementations, OAL 210 includes extensions to document management or content management that allow business objects to use the functionality for documents.

[0058] Service layer 220 provides services for business objects in layer 210. In general, services for business objects are common procedures that users need to interact effectively with the objects. Service layer 220, for example, may include generic services, collaboration services, guided procedure services, and/or a container for application services. By separating the services from the business objects, the services may be more readily reused across business objects.

[0059] UI layer 230 provides user interfaces that allow a user to interact with composite applications. In particular implementations, UI layer 230 provides pattern components, such as, for example, a dashboard, a search bar, a browse and collect function, an object editor, and phases for a guided procedure, as building blocks for user interfaces. UI layer 230 may also

decouple application logic from the UI. As shown, UI layer 230 accomplishes this by having a separation of the business objects, which are in the object access layer 210, and application services, which are in service layer 220, from the user interface elements, which are in UI layer 230. This allows UI components to be reused in different application contexts. This also allows business objects and application services to be visualized differently according to the specific equipments of a certain use case. UI layer 230 may also leverage the metadata information on business objects and services through metadata-driven UI-generation and configuration. The metadata approach allows for ready adaptability to alternative screens depending on the end users needs (e.g., in different industries). UI layer 230 may additionally allow integration (e.g., binding) into OAL 210 to access business objects, business services, and metadata. Thus, UI components may be connected to business objects in OAL 210. UI layer 230 may support any appropriate type of user interfaces, such as, for example, a user interface composed of pattern-based components and/or freestyle components with interfaces to the user interface components -- this user interface will discussed in more detail below -- or Java Server Pages (JSPs) from Sun Java Server Pages (JSPs) from Sun.

[0060] Metadata repository 240 stores the content of the composite application (e.g., specific business objects, information about services, and, eventually, processes) and makes the metadata information available at run-time, if needed. The repository may allow different metamodels to be created (the model for business objects being one of them) and to persist the metadata.

[0061] As mentioned previously, attached to framework 200 is application database 250. Database 250 provides a central repository for available business objects. An example of data

in repository 250 includes database tables for a business object. The data may be added to, changed, and/or deleted. Data may also be stored in KW, BW, or an XI system. As discussed, framework 200 provides a set of standard services that enables application developers to make use of the data. The origin of the data and/or its persistency may be transparent to the application developer, not to mention the composite application.

[0062] Based on the central repository for objects, metadata data about objects is stored in metadata repository 240. This metadata enables generic services like automatic generation of default UIs, object access interface, data access methods, persistency, and mappings.

[0063] Framework 200 may be implemented using readily available technology. For example, the framework may be implemented using mySAP technology components. In particular implementations, the components may include an SAP Web Application Sever (WAS) to run the applications, an SAP Enterprise Portal to render the applications, an SAP KW to handle unstructured information sources, pattern-based components and/or freestyle components with interfaces to the UI components to design UIs and to provide J2EE and ABAP run-time integration, an SAP BW to provide reporting and analytics, data mining, and planning and simulation, SAP Business Process Management (BPM), an SAP Exchange Infrastructure (XI) to provide shared integration knowledge separate from applications, and SAP Web services to offer business functionality over the Internet.

[0064] In one general aspect, framework 200 allows composite applications to work with existing system landscapes. The framework accomplishes this by decoupling composite applications from the underlying enterprise platform, which includes enterprise base systems. This decoupling may involve providing communication to back-end systems via a central

interface and providing a back-end-independent object model. The latter may be implemented so that the data from the source systems may be transformed into a unified structure. This may also allow successive installation, activation, and use of different applications, which may reduce entry costs.

[0065] Examples of the types of business processes supported by the framework including enterprise change management, product innovation, employee productivity, and enterprise service automation. Enterprise change management may support enterprises when merging, splitting, acquiring, spinning off, or reorganizing. Product innovation may support the life cycle of a product, including the prenatal phase of collecting ideas and consolidating them into concepts, the market launch phase, and the end of life. In doing so, the resources of a PLM and CRM may be drawn upon. Employee productivity aims to increase employee productivity, decrease costs, and increase employee satisfaction. Key functions may include manager self services, employee self services, expert finders, e-procurement, and e-learning. ERM and B2E resources may be drawn upon to accomplish these tasks. Enterprise service automation provides administration and monitoring functions as well as evaluation tools to facilitate project success. An example of this is the setting up of projects and the staffing with people with the required skills and availability. Additional application families may also be created.

[0066] FIG. 3 is a block diagram of a composite application framework 300 illustrating details of one potential implementation. As illustrated, framework 300 includes design-time components 310, run-time components 320, and a metadata repository 360, which is shared by the design-time components and the run-time components. In general, design-time components

310 are responsible for developing composite applications that are executed by run-time components 320.

[0067] In more detail, design-time components 310 provide a repository and user interface for modeling and generating business objects, business services, business processes, user interfaces, and/or any other appropriate portions of a composite application. A business object, for example, may be an employee, a product, a plant, or any other semantic representation of a real-world entity. A business service is an action taken on a business object. Changing the price or category of a product are examples of services for a business object that represents a product. As another example, gathering input from employees and customers, who may themselves be represented by business objects, for a new product idea are examples of business services. Putting services together in a proper combination produces a business process. A composite application is typically composed of business objects, business services, and/or business processes.

[0068] As illustrated, design-time components 310 include application modeling tools 312, application generators 314, and, in part, metadata repository 360. Modeling tools 312 may be used for modeling business objects, business services, business processes, user interfaces, and the like. A separate modeling tool may be used for each of the composite application portions. Furthermore, modeling tools 312 may be used for integrating business objects, business services, business processes, user interfaces, and the like. Thus, framework 300 may support model-driven composition of composite applications, allowing for development with little or no programming effort.

[0069] The metadata about business objects, business services, business processes, and/or other application portions is stored in metadata repository 360. Thus, an application portion may be modeled as well as the origin of the data, whether in a local database, remote database, or mixture. In particular implementations, the content of metadata repository 360 may not be associated with a specific implementation of the things a repository describes. Thus, other repositories that describe a specific implementation may be filled from repository 360.

[0070] Generators 314 are used for generating actual code from the portions modeled by modeling tools 312. To accomplish this, the generators may use templates that are stored in metadata repository 360. Driven by the metadata in repository 360, the generators may automatically create Java classes (e.g., for use in run-time components 320) and also configuration files (e.g., to adjust UI patterns to a certain business object). Thus, the connectivity to back-end systems and the application persistency may be generated, as well as a default user interface. The generators may also generate interfaces for application services, data access logic, and persistency.

[0071] Run-time components 320 provide the run-time environment for business objects, business services, business processes, user interfaces, and the like, as well as data access abstraction. As illustrated, run-time components 320 include an object access layer 330, a service layer 340, a UI layer 350, and, in part, metadata repository 360. Run-time components 320 also use an application database 370, which stores data tables for executing applications.

[0072] Object access layer 330 manages interaction between composite applications and enterprise base systems 390. In doing so, layer 330 reduces the knowledge needed for the application developer about the source of data because layer 330 sits on top of and embraces

different connectivity technologies. Thus, layer 330 provides a uniform interface for composite applications. As such, layer 330 may act as a dispatcher to provide access to a variety of data sources. As illustrated, layer 330 leverages a message-based platform 390a that includes an XI with connectivity to underlying applications like CRM, HCM, and PLM, a knowledge management warehouse (KW) 390b, and a business intelligence warehouse (BW) 390c, and manages the persistency in application database 370. Layer 330 may also leverage a fairly synchronous infrastructure such as a service-oriented data access, which could be a BW, and a KW repository framework, which may allow connection to document management systems or to LDAP (Lightweight Directory Access Protocol), a more unstructured type of data. Thus, layer 330 may bring structured and unstructured data closer together. Coding and configuration data for layer 330 may be automatically generated, at least in part, by business object metadata in repository 360.

[0073] For data access abstraction, the fact that layer 330 sits on top of and embraces different connectivity technologies allows routing to a variety of different data sources. Furthermore, layer 330 allows for local persistency (e.g., connectivity to a local database such as application database 370 to store data). Additionally, data synchronization and replication of remote data (e.g., data in back-end systems) into the local persistency database may be supported. The data may be transferred and transformed into the local persistency. For an application sitting on top of layer 330, the source of the data may be completely transparent. For example, some applications underlying layer 330 might provide services (e.g., Web services), allowing the data to be read remotely, and some applications may not provide such access. Layer 330 may deal with this situation by making everything look like a service to the

composite application, perhaps by replicating data into the composite application and offering a local service. Layer 330 may also provide namespace concepts that are used for generated mappings.

[0074] In certain implementations, layer 330 includes extensions to document management or content management that allow business objects to use the functionality for documents. For example, taxonomies for business objects, transparent indexing of TREX for structured and unstructured objects, and subscription services for dependent objects independent of the repository where the objects reside may be provided. Layer 330 may also provide transaction support, in as far as the transaction concept is also supported by concerned source systems, a metadata interface, allowing an application to be dynamically configured at run-time, and subscription services (e.g., J2EE publish and subscribe).

[0075] Layer 330 may have a variety of features. For example, it may keep the application logic and UI stable. This may be accomplished by making the origin of the data transparent to the application logic and UI. Thus, there may be little to no impact of the underlying IT system landscape on the application logic or UI, as the adaptation to a specific landscape is, at least for the most part, accomplished by layer 330. Furthermore, changes to enterprise base systems 390, such as KW and XI, may have no direct influence on the application logic or the UI.

[0076] As another example, layer 330 may accelerate composite application development. This may be accomplished by reusing business objects across composite applications, reusing enterprise base systems 390 access services (e.g., KW and XI) across composite applications, reusing know-how (e.g., uniform interface structure providing common access to business objects), efficient (e.g., model-driven) implementation of business objects based on a

repository, and using a relative homogenous structure for application logic, which simplifies modifications and maintenance.

[0077] As an additional example, layer 330 may enable integration. This may be facilitated by communication between composite applications via a shared object model, shared contexts across composite applications based on a shared object model, and integration of enterprise base system (e.g., KW and BW) via a composite application object model. The integration may also involve integrating business objects.

[0078] As a further example, layer 330 may facilitate application building by configuration. This may be accomplished by providing standard interfaces with well-defined semantics, which allows components to be combined in a meaningful way since the semantics of the components' interfaces is known, and allowing objects to participate in a collaborative context, (e.g., chat room) just by implementing certain interfaces.

[0079] Service layer 340 provides services for business objects in layer 330. In general, services for business objects are common procedures that users need to interact effectively with the objects. A service layer may also provide other types of services, such as, for example, UI-related service and/or collaboration services. Furthermore the service layer may provide integration of external services.

[0080] As illustrated, service layer 340 includes generic services 342, collaboration services 344, guided procedure services 346, and a container for application services 348. By separating the services from the business objects, the services may be more readily reused across business objects.

[0081] As its name implies, generic services 342 provides a set of standard services for parts of an application. Moreover, the services may be used across applications. Generic services 342 may also provide namespace and packaging concepts. The services are typically not bound to a portion of an application, but are available to all portions. Examples of generic services include print services, value help services, authorization, personalization, and voice enablement. An example of a value help service is the filling of drop down boxes in user interfaces; the service is able to determine what the possible entries are for boxes and to populate the boxes therewith.

[0082] Collaboration services 344 provides the ability to link semantic objects to business objects. Semantic objects typically provide a set of generic services, like classification, notification, subscription, feedback, search, retrieval, rating, time-based publishing, state-based publishing, and security model. In addition, relations between semantic objects may be supported. For example, a team could be assigned to a task, and people could be assigned to the team. Moreover, a room could be created for that task, to keep people and documents together. Semantic objects such as document, folder, room, task, meeting, user, and discussion may be accessible via layer 330. Semantic objects may also be available in a variety of other ways. For example, semantic objects may be included in layer 330 as business objects, and/or individual services of semantic objects may be included in layer 340.

[0083] Collaboration services 344 extends the semantic object concept by making the functionality of semantic objects available for business objects (e.g., notification, subscription, etc.). Thus, services 344 provides collaboration context for a business object. Services 344 may automatically manage the relations between business objects and semantic objects. In

addition, new kinds of relations may be supported: for example, relations between business objects and semantic objects. Thus, a task or a team may be assigned to a specific product, people may be assigned to the task, and so on. Furthermore, special collaborative services may be provided for semantic objects, such as scheduling and assignment functions for tasks and inviting, splitting, and closing functions for discussions. In particular implementations, a suite of collaboration services may be provided without the need to deal with KM specific. These services may also be made available for composition applications. Furthermore, the relation between the business objects and the semantic objects may be maintained.

[0084] The collaboration provided by collaboration services 344 may be semi-structured processes. A common understanding of a business process may be reflected by a predefined collaboration scenario. On the other hand, the business process may be adaptable to different enterprise's needs. To support this, differing scenarios may be built with minimal programming.

[0085] Guided procedure services 346 allows business objects to participate in guided procedures. A guided procedure is a series of steps, often involving human interaction, that should be performed during the execution of a composite application. A guided procedure, which is a type of workflow, is typically one that is common to a variety of applications and, thus, may be reused. To provide guided procedures, services 346 may provide pre-defined building blocks for process workflow and pre-defined actions. Additionally, services 346 may facilitate template design. This may be used to support role-based collaborative processes, process workflow, and/or context definition. At run-time, guided procedures may be implemented by using template instantiation, by design-time integration for ad hoc adaptation

of templates, and procedural navigation and integration in a Universal Worklist (UWL). Furthermore, services 346 may provide context awareness and sharing by context mapping of building blocks, business object integration, and user assistance. Additionally, services 346 may provide monitoring and analysis of guided procedures.

[0086] Application services container 348 is used to implement model specific services for one or more business applications. Although generic objects, generic services, and/or processes may be generated for an application, some business logic is too specific to be implemented generically. For example, determining the number of vacation days that an employee has may involve determining the number of vacation days the employee is entitled to per year, determining the number of days available based on the employee's service to date for the year, determining how many days the employee has been absent to date for the year, and determining whether to assign those days to vacation days or sick days. Furthermore, if the employee is splitting time between departments, an allocation may need to be made between the two. As another example, an order process at a manufacturer may include obtaining a order, splitting the order into components based on the bill of materials, determining whether each component is in stock, if a component is not in stock, determining where and/or how to procure it, and, if a component must be procured, determining a potential delivery date. The business logic for such tasks may be difficult to model generically, especially across a wide variety of industries. Thus, the logic may have to be individually coded. Container 348 provides interfaces for the code to be used. The interfaces may be generated by the metadata of the service model, but the inner code has to be programmed individually. Also, maintaining the service definition in the design-time allows generation of an empty service.

[0087] UI layer 350 includes a UI framework 352. Framework 352 provides pattern components as building blocks for user interfaces. Examples of pattern components include a dashboard, a search bar, a browse and collect function, an object editor, and phases for a guided procedure. These components may serve to increase efficiency of UI development because they are reusable and may serve to keep training costs down by providing a standard look and feel for the composite applications. Furthermore, the components readily provide a UI for composite application objects and services and allow a default UI to be automatically generated for displaying, creating, and changing business objects.

[0088] Framework 352 may also decouple application logic from the UI. As shown, framework 352 accomplishes this by having a separation of the business objects and application services from the user interface elements. This allows UI components to be reused in different application contexts. This also allows business objects and application services to be visualized differently according to the specific equipments of a certain use case.

[0089] UI framework 352 may also leverage the metadata information on business objects and services through metadata-driven UI-generation and configuration. The metadata approach allows for ready adaptability to alternative screens depending on the end users needs (e.g., in different industries). UI framework 352 may additionally allow integration (e.g., binding) into layer 330 to access business objects, business services, and metadata. Thus, UI components such as patterns and freestyle may be connected to business objects in layer 330. In accomplishing this, framework 352 may provide the necessary metadata at design-time and manage the access to the according service providers at run-time.

[0090] UI framework 352 may support any appropriate type of user interfaces. For example, the UI framework may support a user interface composed of pattern-based components and/or freestyle components with interfaces to the user interface components -- this user interface will discussed in more detail below -- or Java Server Pages (JSPs) from Sun. UI framework 352 may also support a Java front-end and ABAP back-end, a Java front-end and Java back-end, or any other appropriate combination of front-end and back-end. The framework may additionally provide a construction kit for complex components and applications and configuration of patterns via XML, URL, or other appropriate technique.

[0091] Metadata repository 360 stores the content of the composite application (e.g., specific business objects, information about services, and, eventually, processes) and makes the metadata information available at run-time (if needed). The repository may allow different metamodels to be created (the model for business objects being one of them) and to persist the metadata. For specific purposes, additional repositories, such as, for example, a portal content directory (PCD), which may contain portal specific pieces of an application (e.g., views, pages, roles), may be required.

[0092] As mentioned previously, attached to framework 320 is an application database 370. Database 370 provides a central repository for available business objects. An example of data in repository 370 includes database tables for a business object. The data may be added to, changed, and/or deleted. Data may also be stored in KW, BW, or an XI system. As discussed, framework 200 provides a set of standard services that enables application developers to make use of the data. The origin of the data and/or its persistency may be transparent to the application developer, not to mention the composite application.

[0093] Based on the central repository for objects, metadata data about objects is stored in metadata repository 360. This metadata enables generic services like automatic generation of default UIs, object access interface, data access methods, persistency, and mappings.

[0094] In particular implementations, modelers 312 and generators 314 generate the business objects used in layer 330. Modelers 312 and generators 314 also facilitate the creation of business object metadata and its storage in metadata repository 360. The modelers and generators may be relatively easy to use because they are restricted to a particular purpose. Additionally, they may help to ensure the consistency of the metadata according to the capabilities of layer 330.

[0095] In particular implementations, the composite application portions may be implemented as Enterprise Java Beans (EJBs). In other implementations, the design-time components may have the ability to generate the run-time implementation into different platforms, such as J2EE, ABAP, or .NET.

[0096] Components 310 may also support a variety of specific features needed for business objects, such as time-dependent attributes or organizational-unit-dependent attributes, like product attributes, which differ from plant to plant. The components may not only generate the classes and the coding, but may also create the database tables and the interfaces to the UI, including the relevant metadata. So, after modeling, there may be a generation step that provides a stack of services for one business object, including the UI down to the database tables and proxies for remote access.

[0097] In particular implementations, generators 314 allow template-based generation of Java-coding, database tables, entries in metadata repository 360, XML configuration files, etc.

This may be implemented with extensibility and the ability to conduct upgrades without loosing his information. This capability may be achieved by allowing the metadata of the new implementation to be compared with the metadata of the existing implementation during an upgrade. If there are implementation-specific extensions, they may be identified, and strategies for solution of possible conflicts may be proposed.

[0098] As discussed, framework 300 provides modeling and configuration tools (e.g., business object modelers and guided procedures), generic components (e.g., UI patterns and generic services (F1, F4, help, authorizations)), standardized interfaces (e.g., object access layer), reusable content (e.g., predefined object models and XI content), and integration infrastructure (e.g., to connect business objects and documents and access to XI proxies). Moreover, the framework allows composite applications to be created according to guidelines, which are documentation that allows composite applications to be created appropriately using the framework. The guidelines may or may not be implemented in software.

[0099] Framework 300 may be implemented using readily available technology. For example, the framework may be implemented using mySAP technology components. In particular implementations, the components may include an SAP Web Application Sever (WAS) to run the applications, an SAP Enterprise Portal to render the applications, an SAP KW to handle unstructured information sources, an SAP BW to provide reporting and analytics, data mining, and planning and simulation, SAP Business Process Management (BPM), an SAP Exchange Infrastructure (XI) to provide shared integration knowledge separate from applications, and SAP Web services to offer business functionality over the Internet.

[00100] For instance, an SAP WAS may include a J2EE engine, SAP IDE, Universal Workflow, and Deployment Service. The WAS may also include a pattern-based and freestyle-based user interface development and interface module. Also, an SAP Enterprise Portal may provide unified access to applications, information, and services by using views, roles, pages, worksets, top-level navigation, and KM. This enterprise portal also provides login management and user management. For KM, unstructured information consists of collaboration and content management. For collaboration, KM enables team-driven business processes, synchronous and asynchronous applications, groupware integration, calendars, bulletin boards, threaded discussions, and collaboration rooms. For content management, KM handles documents, feedback, rating, publishing, subscription, document workflow, versioning, archiving, indexing, searching, and taxonomies. SAP BPM may cover life cycles (e.g., design, development, deployment, and change). An SAP XI may provide external and internal integration of system and connectors to various systems such as Oracle, Siebel, Peoplesoft, and SAP. The SAP XI may be based on Web services, JAVA, and XML standards. SAP Web services may provide a service provider, service handler, and service user. Additionally, an SAP BW may be used.

[00101] Moreover, the KM and collaboration functionality may embedded in applications, not only in separate pages in the portal. Furthermore, any general development environment may be used. For example, the development environment could include Java, with EJB 2.0, JDO, Java persistency, and Java application logic, Advanced Business Application Programming (ABAP), and Web services. Existing ABAP components may be integrated via Java connector

calls. In particular implementations, the complete Java stack could be used. Furthermore, Web service technology may be used for remote access.

[00102] In one general aspect, framework 300 allows composite applications to work with existing system landscapes. The framework accomplishes this by decoupling composite applications from the underlying enterprise platform. This decoupling may involve providing communication to back-end systems via a central interface and providing a back-end-independent object model. The latter may be implemented so that the data from the source systems may be transformed into a unified structure. This may also allow successive installation, activation, and use of different applications, which may reduce entry costs.

[00103] In another general aspect, the framework facilitates highly efficient development of composite applications. This may be accomplished by model-driven composition of applications, service-oriented architecture, UI and application patterns, reusable object models, and methodologies. Thus, the framework could be viewed as a kind of application factory. In such a factory, application building may be enabled without programming, at least as far as possible.

[00104] Examples of the types of business processes supported by the framework including enterprise change management, product innovation, employee productivity, and enterprise service automation. Enterprise change management may support enterprises when merging, splitting, acquiring, spinning off, or reorganizing. Product innovation may support the life cycle of a product, including the prenatal phase of collecting ideas and consolidating them into concepts, the market launch phase, and the end of life. In doing so, the resources of a PLM and CRM may be drawn upon. Employee productivity aims to increase employee productivity,

decrease costs, and increase employee satisfaction. Key functions may include manager self services, employee self services, expert finders, e-procurement, and e-learning. ERM and B2E resources may be drawn upon to accomplish these tasks. Enterprise service automation provides administration and monitoring functions as well as evaluation tools to facilitate project success. An example of this is the setting up of projects and the staffing with people with the required skills and availability. Additional application families may also be created.

[00105] The framework may also provide external services in a shared object environment. This may be accomplished by the framework providing a uniform object access layer and service layer that bundle functionality across service components. Furthermore, a transparent mapping may be provided from the application's perspective. Thus, the application built on the framework would not have to know whether certain services are provided by a portal, by a KW, by a WAS, or other external service.

[00106] FIG. 4 illustrates design-time components 400 for a composite application framework. Design-time components 400 could be representative of design-time components 310 in FIG. 3.

[00107] As illustrated, design time components 400 include a business object modeler 410, a business object generator 430, and a metadata repository 450. Note that metadata repository 450 is also a run-time component.

[00108] Business object modeler 410 includes an Integrated Development Environment (IDE) application program interface (API) 411, an object modeler 412, and a relation modeler 413. IDE API 412 allows modeler 412 to be integrated into an Eclipse IDE, which supports the modeling of the business object by object modeler 412. For example, the integration supports

generation of business objects as EJBs, interfaces for application services, default user interfaces, data access logic, and persistency. Relation modeler 413 allows the modeling of relations between modeled objects. For example, a sales order could be composed of a customer, a product, and a price. Relation modeler 413, therefore, allows for the modeling of the relations between these items. In operation, for instance, if a user interface is generated for a sales order, the semantics for each field in the sales order may be identified. Additionally, a connection to the value help function may be facilitated.

[00109] Modeler 410 also includes a metadata API 414 and a generation API 415. Metadata API 414 allows object modeler 412 to store and access business object metadata in metadata repository 450 and relation modeler 416 to store and access business object relation metadata in metadata repository 450. Generation API 422 allows a business object to communicate with generator 430 for code generation.

[00110] Generator 430 includes a generator framework 432, a persistency generator 434, an EJB 436 generator, a UI adapter generator 438, a Web service generator 440, and a metadata API 442. Generator framework 432 may also be integrated into the Eclipse IDE.

[00111] To generate a business object, generator 430 may use templates in metadata repository 450 and code them with object metadata and relation metadata in the repository. Generator 430 may also generate the data persistency for the business object, and generate the actual business object, an EJB in this instance. Generator 430 may additionally generate user interfaces for the business object and any necessary Web services.

[00112] The templates may be generic. In particular implementations, the generators automatically create Java classes (e.g., for the implementation of the object access layer), JDO

tables, EJBs, and configuration files, to adjust UI patterns to a certain business object, for example. Thus, the connectivity to back-end systems and the composite application persistency is generated as well as a default User Interface. Furthermore, UI adapters for a UI development and interface module and, if necessary, Web services may be generated. The output of such a process may be real working code in the object access layer of the run-time components.

[00113] One example is the generation of a run-time implementation of a business object in an object access layer. The generator reads the business object metadata from the repository and generates the JDO persistency, the connectivity to the XI, the KW and/or the BW (e.g., by using proxies), the generic methods, and the basic UI. For this coding, templates (e.g., for services) or XML-templates (e.g., for JDO persistency) are used where business object specific coding or XML is added, and the result is stored as complete code or complete XML.

[00114] Metadata repository 450 includes object metadata 452, relation metadata 454, and code generation templates 456. As mentioned previously, the information in object metadata 452 and relation metadata 454 may be used to code templates 456 to generate a business object.

[00115] FIGs. 5A-I illustrate user interfaces 500 for an implementation of a business object modeler. In specific, user interfaces 500 demonstrate how a business object may be created using a framework similar to framework 300 in FIG. 3. As illustrated, user interfaces 500 are graphical user interfaces, but the interfaces may be in any appropriate format and may have any appropriate data and/or arrangement of data.

[00116] FIG. 5A illustrates user interface 500a. User interface 500a demonstrates the creation of a business object for a composite application for defining a product. Creation of business objects for other composite applications could be similar.

[00117] As illustrated, user interface 500a includes a directory section 510 and a work section 520. Directory section 510 includes a listing of business objects for product definition. The creation of a new business object may be initiated by right-clicking in directory section 510, as shown, and selecting the appropriate line in the generated menu. In response to the initiating command, work section 520 includes an association section 522 in which a name is associated with the new business object and the business object is associated with an composition application, "xApp-xPD" in this instance.

[00118] FIG. 5B illustrates user interface 500b. Once a name and composite application is associated with the new business object, user interface 500b allows specification of general information regarding the business object.

[00119] As illustrated, user interface 500b contains directory section 510 and work section 520. Note that directory section 510 has expanded by one object for the new business object (i.e., "Concept"). Work section 520 includes information section 524, which allows specification of general information about the business objects. Section 524 allows the specification of the persistency model (e.g., whether the object has a local persistency or whether it should be read on the fly from a back-end system) and administrative information (e.g., about the owner, status, release, and version of the object).

[00120] FIG. 5C illustrates user interface 500c. Once general information about the business object is specified, user interface 500c allows attributes to be added/deleted from the object.

[00121] As illustrated, user interface 500c includes directory section 510 and work section 520. Work section 520 includes an attribute specification section 526, which allows the specification of attributes. As shown, section 526 includes a first section 527a, in which available attributes are displayed, and a second section 527b, in which the current attributes for the business object being modeled are shown. Section 526 allows attributes to be added/deleted by selection and command techniques, by drag and drop techniques, or by any other appropriate technique. Attributes may be full-blown business objects or objects that exist only in the context of another object. A decision may be made as to whether the values for dependent objects should be shared across objects or not.

[00122] FIG. 5D illustrates user interface 500c with attributes from first section 527a incorporate into second section 527b. Thus, the business object being modeled is being developed.

[00123] FIG. 5E illustrates user interface 500d. Once attributes have been specified for the business object, user interface 500d allows methods to be specified.

[00124] As illustrated, user interface 500d includes directory section 510 and work section 520. Work section 520 includes a method specification section 528, which allows the specification of methods. As shown, section 528 includes a first section 529a, in which methods may be specified, and a second section 529b, in which parameters for the methods may be specified.

[00125] There are a variety of types of methods that may be created for business objects. One example is lifecycle methods (e.g., create, update, etc.). Another example is standard methods that do not require coding. An example of such methods are those that allow business objects

to participate in collaboration (e.g., subscribe, notify, discuss, etc.). These methods may be automatically called when a certain action is carried out (e.g., when a business object should be updated). Another type of method is one that is specific to the composite application. These methods may be directly called by the applications.

[00126] FIG. 5F illustrates user interface 500e. Once methods for business objects have been specified, user interface 500e allows aspects of the business object to be defined. Aspects are, in general, subsets of attributes and/or methods for the business object. Aspects may be used to lay an industry or application specific view on general business objects.

[00127] As illustrated, user interface 500e includes directory section 510 and work section 520. Work section 520 includes an aspect specification section 530, which allows the specification of aspects. As shown, section 530 includes a first section 531a, in which attributes of the business object are displayed. (The attributes were defined in user interface 500c.) Section 530 also includes a second section 531b, in which the attributes for the different aspects of the business object may be specified. Section 530 allows attributes to be added/deleted by selection and command techniques, by drag and drop techniques, or by any other appropriate technique.

[00128] FIG. 5G illustrates user interface 500f. Once aspects for business objects have been specified, user interface 500f allows the specification of technical data for tables if the business object is to have its own local persistency. In this example, the tables are JDO tables, although they could be of any other appropriate type.

[00129] As illustrated, user interface 500f includes directory section 510 and work section 520. Work section 520 includes table specification section 532, which shows technical

information about the generated database tables. An example of technical information is the name of the database table in the data dictionary.

[00130] FIG. 5H illustrates user interface 500g. Once tables, if any, have been specified for a business object, user interface 500g allows the specification of information for proxies, which may be used if data for a business object should be read remotely from a back-end system. For example, an XI could be specified here.

[00131] As illustrated, user interface 500g includes directory section 510 and work section 520. Work section 520 includes proxy specification section 534, which allows the specification of proxies. A connection to an XI could be defined here, as well. As shown section 534 includes a first portion 535a and a second portion 535b. First portion 535a allows the specification of proxies for data actions for the objects. Second portion 535b allows the specification of mappings for attributes.

[00132] FIG. 5I illustrates user interface 500h. Once proxies have been specified, the code may be generated, as discussed previously, and displayed in user interface 500h.

[00133] As illustrated, user interface 500h includes directory section 510 and work section 520. Work section 520 includes code section 536, which allows the generated code to be displayed. Furthermore, the code for methods that have been defined beforehand may be added. In this implementation, the method headers are already generated into the displayed coding. At this stage, the whole run-time environment for the business object could be generated.

[00134] FIG. 6 illustrates run-time components 600 for a composite application framework. Run-time components 600 could be representative of run-time components 320 in FIG. 3.

[00135] The illustrated components 600 are for an object access layer 610. Layer 610 has a variety of business objects 620 executing therein. Each business object 620 includes a business object bean 622 that includes lifecycle methods, properties, and validation. The beans 622 may, for example, be Enterprise Java Beans (2.0) and may be generated by the metadata, as already mentioned. Each business object 620 also includes a data object 624. Data object 624 allows bean 612 to be decoupled from the underlying data, which is accessed through data a access service factory 630.

[00136] Data access service factory 630 is responsible for homogenizing data from various sources. In accomplishing this, factory 630 determines data sources for business objects 620. Thus, it may be viewed as a kind of dispatcher. For example, data access service factory 630 may read data from a remote service 642, a persistence service 644, a KM service 646, and/or an OLAP service 648 and present this data to data object 624. The data access service factory 630 may accomplish this by allowing the plug in of special adapters for accessing different sources of data. The composite application framework may provide adapters for the exchange infrastructure (XI), the knowledge management system (KM), and the data warehouse (BW). It may also provide access to a local database that is treated the same way as remote access adapters.

[00137] Business objects 620 also include interfaces for local and remote data access. The interface may be used for accessing application services, such as those in service layer 340 in FIG. 3, a UI layer, such as UI layer 350 in FIG. 3, or any other appropriate type of data. For the application services and the UI, layer 610 supports three different interfaces — a local interface 626, a Web-service interface 627, and a service provider interface 628. Local interface 626 is

used when no remote access is necessary. Web service interface 627 is used for remote access. Service provider interface 628 is used to access UI patterns.

[00138] Layer 610 also includes metadata access services 650 and object access services 660. Metadata access services 650 provides access to the metadata in metadata repository 690. Object access service 660 provide the elementary access methods (e.g., create, read, update, delete) for business objects.

[00139] In operation, layer 610 manages the access to business objects 620 and manages the connectivity to the underlying applications, whether via XI, the persistency in an application database, the access to KM objects (like documents attached to the business object), and/or to BW, with the ability to replicate data into BW for reporting and analysis issues. Layer 610 also reduces the knowledge needed for the application developer about the source of the object.

[00140] FIGs. 7A-B illustrate components 700 for guided procedures for a composite application framework. As mentioned previously, a guided procedure is a series of steps, often involving human interaction, that should be performed during the execution of a composite application. A guided procedure is typically one that is common to a variety of applications. As illustrated, components 700 may be classified into design-time components 710 and run-time components 750, except for a metadata repository 790, which is part of both. Design-time components 710 may be used to generate run-time components 750.

[00141] Design-time components 710 include a modeler 720 and a generator 730. Modeler 720 includes a process modeler 722, a pattern modeler 725, and an action modeler 726. Process modeler 722 includes a workflow modeler 723 and a context modeler 724. As their names imply, workflow modeler 723 allows process workflow for a guided procedure to be

modeled, and context modeler 724 provides context definition. That is, context modeler 724 allows relations between other processes to be defined. As an example of this, an application may have more than one way of being activated — Intranet Web-based form versus remote voice control, for example. Context modeler 724 is responsible for making sure that both activation mechanisms are associated with the application. Pattern modeler 725 provides workflow patterns (e.g., delegation and approval) for workflow modeler 723, and action modeler 726 provides actions for workflows.

[00142] Modeler 720 also includes a metadata API 727, which provides access to the data in metadata repository 790. Thus, access to meta data regarding guided procedures is available.

[00143] Generator 730 includes a template generator 731, a state chart generator 733, a pattern generator 735, an action generator 737, and a metadata API 739. Templates describe a workflow that may be may be implemented using workflow patterns. Workflow patterns contain actions that must be accomplished to complete the workflow and, hence, part of the template. Thus, a pattern may be viewed as an abstraction of an action, and a template may be viewed as an abstraction of work flow pattern.

[00144] For example, a template could describe a workflow for ordering a product — a computer, for example. The template may specify a workflow pattern for obtaining manager approval. The pattern would have certain actions that need to be undertaken. An example of an action could be finding the names of the employee's managers. The approval pattern, moreover, could be used for different templates.

[00145] As their names imply, template generator 731 generates templates, state chart generator 733 generates state charts, pattern generator 735 generates patterns, and action

generator 737 generates actions for the run-time environment. Metadata API 739 provides access to the metadata in metadata repository 790.

[00146] Metadata repository 790 includes templates 792, workflow patterns 794, actions 796, and metadata 798. The templates, patterns, actions, and metadata may be accessed by generator 730 to produce a guide procedure.

[00147] Run-time components 750 provide instantiation for guided procedures, producing instances 752. Procedural navigation and integration may be provided in a Universal Worklist (UWL).

[00148] Run-time components 750 also include object access services 760, context sharing service 762, content services 764, portal connector service 766, KM service 768, workflow service 770, and metadata services 772. Object access services 760 allows objects in an object access layer to be accessed. Context sharing service 762 provides context to a workflow. For example, when a user accesses a workflow, context sharing service 762 provides a link to the proper portions of the workflow. For instance, many workflows involve inboxes, where new tasks for the workflow may be sent. The inbox may provide a link to the proper portion of the workflow if the context is known. Content services 764 provide services for executing functions based on generic calls. For example, a workflow may need an application — a composite application, an HRM application, or a CRM application, for example — to be initiated. By making a generic call to content services 764, the application may be initiated. Content service 764 may support integration with an application and/or a user interface. Portal connector service 766 provides a connection service to a portal. KM service 768 provides a connection service to a KM module. Workflow service 770 provides a connection service to

an ad-hoc workflow. This workflow may be very user-centric, allowing the assignment of not only tasks handled by transactions in business systems, but also tasks that require user handling (e.g., compose e-mail). Metadata services 772 provides a connection to metadata repository 790.

[00149] Components 700 may have a variety of features. For example, the components may provide context mapping for building blocks, and a user profile may be automatically used and updated. In certain implementations, ad-hoc administrations of running workflows may be supported and guided procedures may be monitored and analyzed.

[00150] FIGs. 8A-K are user interfaces 800 that illustrate the implementation of a guided procedure. In particular, user interfaces 800 demonstrate how a composite application having a guided procedure may be executed in a framework similar to framework 300 in FIG. 3. In this example, the guided procedure is for the hiring of a new employee. Other guided procedures may have similar characteristics. Furthermore, other user interfaces may be in any appropriate format and may have any appropriate data and/or arrangement of data.

[00151] FIG. 8A illustrates user interface 800a. User interface 800a demonstrates the initial view that a user may be shown upon accessing an enterprise management system. User interface 800a contains a first portion 802, a second portion 804, and a third portion 806. First portion 802 lists the activities of the user. One of those activities involves guided procedures. Second portion 804 lists the user's collaborative activities, which were discussed previously. Third portion 806 is a table containing the guided procedures with which the user is associated. Thus, the current guided procedures for the user include preparing for meetings, hiring

employees, organizing workshops, and preparing for trade shows. Portion 806 may be displayed by selecting the guided procedure activity in portion 802.

[00152] FIG. 8B illustrates user interface 800b. User interface 800b demonstrates the status of the second guided procedure in portion 806 of user interface 800a. User interface 800b may be displayed by selecting the second guided procedure in portion 806.

[00153] As illustrated, user interface 800b has a first portion 810, a second portion 812, a third portion 814, and a fourth portion 816. The illustrated guided procedure has a variety of steps, which portion 810 illustrates by the use of a procedure line. Currently, the employee hiring process has proceeded from initialization to hiring to preparing workspace, and is waiting for preparation of the collaborative environment for the new employee. Portion 812 lists the tasks to accomplish the current step, and portion 814 provides more detailed information regarding the tasks. Portion 816 lists resources, in this case people, for the user.

[00154] FIG. 8C illustrates user interface 800c. User interface 800c demonstrates the selection of a mentor for the new employee, the first task in the step. User interface 800c may be displayed upon the selection of the first task in portion 812 of user interface 800b.

[00155] As illustrated, user interface 800c include portion 810 and portion 816 as in user interface 800b. User interface 800c also includes a portion 820, which displays information regarding a selected task. The selected task is shown as highlighted in portion 810. Information for accomplishing the task appears in portion 820. In this illustration, a mentor may be selected and an appropriate message entered.

[00156] FIG. 8D illustrates user interface 800c at another next stage of the mentor selection process. As before, user interface 800c includes portion 810, portion 816, and portion 820. At

this stage, though, a person named "Markus" has been selected in portion 816 to be the mentor for the new employee. Note that the list of people in portion 816 may be modified by altering the selected category "<people>" or by performing a narrowing search. After selecting the appropriate person, the selection may be finalized by the activation of a button in portion 816.

[00157] FIG. 8E illustrates user interface 800c at still another stage of the mentor selection process. As before, user interface 800c includes portion 812, portion 816, and portion 820. But at this stage, the name selected in portion 816 has been finalized. Thus, the name is populated into portion 820. Furthermore, a user has entered a text message to be sent to the selected mentor. Once any message has been finalized, the mentor selection process may be completed by activating a button in portion 820.

[00158] FIG. 8F illustrates user interface 800d. User interface 800d illustrates another task in this step of the new employee guided procedure. In this step, colleagues of the new employee are to be identified and notified of the new employee.

[00159] User interface 800d includes portion 810, portion 816, and a portion 824. In portion 810, another task in the guided procedure step has been selected. Note that the first task in the step has an indicia signifying that it has been completed. The new selection leads to the display of portion 824, in which colleagues may be informed of the new hire. These people may be selected in portion 816.

[00160] FIG. 8G illustrates user interface 800d at another next stage of the colleague identification and notification task. As before, user interface 800d includes portion 810, portion 816, and portion 824. At this stage, however, the colleagues to be notified about the

new employee have been selected from a list of teams in portion 816. After selecting the appropriate team, the selection may be finalized by the activation of a button in portion 816.

[00161] FIG. 8H illustrates user interface 800d at another next stage of the colleague identification and notification task. As before, user interface 800d includes portion 810, portion 816, and portion 824. But at this stage, the colleagues have been selected and finalized in portion 816. Thus, they appear in portion 824. Additionally, a message to the selected colleagues has been entered in portion 824 by a user. Once any message has been entered, the mentor selection process may be completed by activating a button in portion 824.

[00162] FIG. 8I illustrates user interface 800e. User interface 800e illustrates another task in the fourth step of the new employee guided procedure. In this step, a location for the new employee is to be selected.

[00163] User interface 800e includes portion 810, portion 816, and a portion 828. In portion 810, another task in the fourth guided procedure step has been selected. Note that the first two tasks in the step now have indicia signifying that they have been completed. The new selection leads to the display of portion 828, in which a location for the new employee may be specified.

[00164] FIG. 8J illustrates user interface 800e at another stage of the location task. As before, user interface 800e includes portion 810, portion 816, and portion 828. But here, the user resources displayed in portion 816 are the rooms available for assignment by the user. Note that the displayed information category is "<rooms>." Furthermore, the user has selected one of the rooms in the list for the new employee. The selection may be finalized by activating a button in portion 816.

[00165] FIG. 8K illustrates user interface 800e at yet another stage of the location task. Now, user interface 800e illustrates the result of finalizing the location selection. Thus, the location for the new employee is displayed in portion 828. Furthermore, the user is presented with further specification options (i.e., set assignments). The user may then select between the further options and finalize the task, and, hence, the guided procedure step.

[00166] After user interface 800e, the fourth step in the new employee guided procedure is complete. The guided procedure would then move to the fifth step — "Prepare Entry." Now, if the user were to access the procedure again, the procedure line illustrated in portion 810 in FIG. 8B would be updated for the new employee guided procedure.

[00167] FIG. 9 is a user interface 900 illustrating collaboration services. As illustrated, user interface 900 demonstrates a semantic object regarding user tasks. Thus, tasks 910 for a user are displayed in user interface 900.

[00168] Furthermore, user interface 900 contains information regarding the status of tasks 910. For example, user interface 900 indicates task completion time 920, task priority 930, task assignee 940, and task progress 950. These tasks may be related to business objects.

[00169] Other collaboration services that may be provided for tasks could include notification (e.g., when a task is completed or due), feedback (e.g., when a task is complete), and team assignment. In fact, it is possible that assignees 940 are part of a team working on the task.

[00170] FIG. 10 illustrates a user interface framework 1000 for a composite application framework. Framework 1000 may be similar to framework 352 in FIG. 3. In general, framework 1000 allows a user interface to be implemented for a business object.

[00171] As illustrated, framework 1000 includes a business object service module 1010, a pattern configuration module 1020, a metadata adapter 1030, and an EJB adapter 1040. In general, service module 1010, metadata adapter 1030, and EJB adapter 1040 are responsible for translating data into a format usable by pattern configuration module 1020. This makes a separation between the components of framework 1000 and an object access layer 1080, an example of which was discussed previously. Service module 1010 may be an API that allows business object data to be brought into the UI framework environment. Service module 1010 includes a generic client proxy 1012, which prepares data for other components of the UI framework 1000 and/or for external UI components. Proxy 1012 may transform data to a generic API. In effect, then, it may wrap an API to provide an API for the upper-level components. Metadata adapter 1030 allows the metadata of a business object model to be exposed to the UI world, so that a unified metadata repository exists along the whole stack. EJB adapter 1040 allows EJB components, such as tables, business objects, and data, to be translated into a format for framework 1000. Pattern configuration module 1020 contains the configured patterns generated using design-time components.

[00172] As illustrated, pattern configuration module 1020 and generic client proxy 1012 interact with elements of a UI development and interface module 1090. Module 1090 may provide a run-time environment plus a set of development tools based on the Model View Controller (MVC) paradigm. The tools support the selection of patterns by the application developer and/or allow a Web developer of enterprise applications to freely define the layout and flow of an application.

[00173] The technology of module 1090 may be based on JavaServer Pages (JSPs) and a tag library containing, among other things, ready-made user interface elements (such as push buttons and input fields). The description of the application in a metadata repository is used to generate the run-time code for one of the following run-time environments: Java, ABAP, or .NET.

[00174] In more detail, module 1090 provides a user the graphical and declarative development tools needed to design a data model and user interfaces for a Web application. So, for example, a user may simply define input validation, input help, and error handling, and all the required run-time objects may be generated automatically.

[00175] That means the majority of the code is generic and, therefore, is available for each run-time. It also means that only that fraction of the programming that is specific to the run-time, like eventing, needs to be coded.

[00176] Other features may also be included -- for example, a usability pattern-based development approach, the ability to convert a module-based transaction into a Web application, and a client-side framework that keeps the context of a running application and guarantees flicker-free screens. With such features a UI module-based application may consist of a set of views, navigation between the views, concepts for managing the views and determining their sequence, a context for keeping session data, and the business logic of the application. The Model View Controller paradigm allows for a strict separation of presentation logic, business logic, navigation, and eventing.

[00177] UI module developers may also choose to use a freestyle Web development, where controls of the control library may be freely placed onto views according to their respective

company style or a strict pattern-based approach. The latter allows usability engineers to predefine reusable patterns common to a set of business processes.

[00178] UI module tools that enable a developer to create such applications are part of the integrated development environment (IDE) of the SAP Web Application Server. The design tools consist of a view designer, editors for controllers, an application modeler to define the flow of an application, and a model designer.

[00179] UI module may also support location-independent rendering. That is, depending on the capabilities of the client, the module run-time will generate the HTML for the browser either on the client or on the server. Client-side rendering, using the JavaScript-based client-side framework will be possible for Microsoft Internet Explorer as of version 5.5 (or higher) and for higher versions of Netscape Navigator.

[00180] Earlier versions of Microsoft Internet Explorer or Netscape Navigator, as well as a variety of mobile devices, will be supplied with output that is rendered on the server. Server-side rendering is also the only choice for browsers where JavaScript has been disabled for security reasons.

[00181] The declarative separation of the user interface and the data model make it possible to isolate the layout (HTML) from the data (XML). This in turn results in efficient caching, a lower transmission bandwidth requirement, flicker-free screen updates, and virtually device-independent development for browsers, PDAs, and WAP-enabled cell phones. The Web UI module technology may automatically provide support for multiple languages, personalization, and interface customization.

[00182] The separation of the business object and application services from the UI elements allows UI components to be reused in different application contexts and business object and application services to be visualized differently according to the specific requirements of a certain use case. Thus, application implementation may be decoupled from UI technology.

[00183] The configured patterns in pattern configuration module 1020 may be parameterizations of patterns for specific uses. For example, a pattern could specify the lay-out of information on a screen. For instance, a pattern could specify that a list of items could be in a top portion with detailed information of a highlighted item in a bottom portion. This is commonly seen in e-mail presentation modules where a list of new e-mails is on top and a preview screen for a highlighted e-mail is on the bottom. The same pattern, however, could have other uses, such as, for example, in a file manager.

[00184] An example of a user interface for a composite application is illustrated by FIG. 9. As illustrated therein, there is a pattern to list projects in the first column and data about the projects in additional columns. In this UI, however, context is provided for the projects. For example, the relationship between the listed project and the underlying business object in the object access layer is maintained. Thus, selecting an item in the UI may produce more information about the item.

[00185] As another example, a dashboard may be created for a business object using a configuration for a dashboard pattern in the UI pattern development environment such as the UI module development environment. First, the business object would be selected. Then, one of the aspects of the business object would be selected. As discussed previously, an aspect is a subset of attributes for a business object. The attributes that are part of the selected aspect may

then be assigned to column categories. Furthermore, attributes may be suppressed. Additionally, whether there is a hierarchy in the dashboard and how many rows should be displayed may be specified. A preview of the dashboard may then be had after completing the adjustments.

[00186] As discussed, a composite application framework may extend an underlying enterprise platform. For example, a composite application framework may extend my SAP Technology by adding modeling and configuration tools (e.g., business object modeler and guided procedures). As another example, generic components (e.g., UI patterns and generic services) may be included. As a further example, a standardized interface (e.g., object access layer) may be used to access the enterprise base systems. As an additional example, a framework may provide reusable content (e.g., predefined object models and XI content), and integration infrastructure (e.g., to connect business objects and documents and to provide access to XI proxies). Furthermore a composite application framework may provide guidelines (e.g., composite application cookbook). The guidelines may be designed to work cooperatively with guided procedures.

[00187] Although discussed primarily in the context of business applications, it should be recognized that the framework could be used to overlay any collection of enterprise components, whether for a business purpose or not.

[00188] Various implementations of the systems and techniques described here may be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations may include implementation in one or

more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

[00189] These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and may be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the term "machine-readable medium" refers to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term "machine-readable signal" refers to any signal used to provide machine instructions and/or data to a programmable processor.

[00190] To provide for interaction with a user, the systems and techniques described here may be implemented on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user may provide input to the computer. Other kinds of devices may be used to provide for interaction with a user as well; for example, feedback provided to the user may be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user may be received in any form, including acoustic, speech, or tactile input.

[00191] The systems and techniques described here may be implemented in a computing system that includes a back-end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front-end component (e.g., a client computer having a graphical user interface or a Web browser through which a user may interact with an implementation of the systems and techniques described here), or any combination of such back-end, middleware, or front-end components. The components of the system may be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

[00192] The computing system may include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[00193] Although only a few embodiments have been described in detail above, other modifications are possible. In certain implementations, multitasking and parallel processing may be preferable.

[00194] Other embodiments may be within the scope of the following claims.